D1

a processor configured to use the interpreter to interpret an application derived from a Java application program for execution, wherein the application is derived by converting the Java application program into the subset.

91. (Thrice Amended) An integrated circuit for use with a terminal, comprising:

a communicator configured to communicate with the terminal;

D2

a memory storing a first application that has been processed from a second application written in the JAVA programming language or a subset of the JAVA programming language and having a plurality of language elements including at least one string of characters, the string of characters being replaced in the first application by an identifier; and

a processor coupled to the memory, the processor configured to use an interpreter to interpret the first application for execution and to use the communicator to communicate with the terminal.

93. (Thrice Amended) A method for use with an integrated circuit card and a terminal comprising:

D3

processing a [second] Java language application to create a [first] a derivative application, the [second] Java language application having at least one programming element being a string of characters;

replacing the string of characters of the [first]Java language application with an identifier in the [second]derivative application;

storing an interpreter and the [first]derivative application in a memory of the integrated circuit card; and

using a processor of the integrated circuit card to use the interpreter to interpret the [first]derivative application for execution.

65
~~106~~. (Twice Amended) A microcontroller having a set of resource constraints and comprising:

a memory, and

an interpreter loaded in memory and operable within the set of resource constraints,

the microcontroller having: at least one application loaded in the memory to be interpreted by the interpreter, wherein the at least one application is generated by a programming environment comprising:

a) a compiler for compiling application source programs written in high level language source code form into a compiled form, and

b) a converter for post processing the compiled form into a minimized form suitable for interpretation within the set of resource constraints by the interpreter, wherein the converter comprises means for translating from the byte codes in the compiled form to byte codes in a format suitable for interpretation by the interpreter by:

a) using at least one step in a process including the steps:

a.1) recording all jumps and their destinations in the original byte codes;

a.2) converting specific byte codes into equivalent generic byte codes or vice-versa;

a.3) modifying byte code operands from references using identifying strings to references using unique identifiers; and

a.4) renumbering byte codes in the compiled form to equivalent byte codes in the format suitable for interpretation; and

b) relinking jumps for which destination address is effected by conversion step a.1, a.2, a.3, or a.4.

D14

Please cancel Claim 113 without prejudice.

78
~~120~~. (Twice Amended) A method of programming a microcontroller having a memory and a processor operating according to a set of resource constraints, the method comprising the steps of:

D5

63

D

inputting an application program in a first programming language;

compiling the application program in the first programming language into a first intermediate code associated with the first programming language, wherein the first intermediate code being interpretable by at least one first intermediate code virtual machine;

converting the first intermediate code into a second intermediate code, wherein the step of converting comprises:

at least one of the steps of:

a) recording all jumps and their destinations in the original byte codes;

b) converting specific byte codes into equivalent generic byte codes or vice-versa;

c) modifying byte code operands from references using identifying strings to references using unique identifiers; and

d) renumbering byte codes in a compiled format to equivalent byte codes in a format suitable for interpretation; and

relinking jumps for which destination address is effected by conversion step a), b), c), or d);

wherein the second intermediate code is interpretable within the set of resource constraints by at least one second intermediate code virtual machine; and

loading the second intermediate code into the memory of the microcontroller.

D5

Please cancel Claim 123 without prejudice.

127. (Amended) A microcontroller operable to execute derivative programs which are derivatives of programs written in [an interpretable] the Java programming language, the microcontroller having a memory and an interpreter, [the microcontroller] and comprising:

D6

(a) the microcontroller operating within a set of resource constraints including the memory being of insufficient size to permit interpretation of programs written in the [interpretable]Java programming language; and

(b) the memory containing an interpreter operable to interpret the derivative programs written in the derivative of the [interpretable]Java language wherein a derivative

64

of a program written in the [interpretable]Java programming language is derived from the program written in the [interpretable]Java programming language by applying at least one rule selected from a set of rules including:

(1) mapping strings to identifiers;

(2) performing security checks prior to or during interpretation;

(3) performing structural checks prior to or during interpretation; and

(4) performing semantic checks prior to or during interpretation.

84
~~137~~. (Amended) An integrated circuit card for use with a terminal, comprising:

a communicator configured to communicate with the terminal;

a memory storing:

an application derived from a program written in a high level programming language format wherein the application is derived from a program written in a high level programming language format by first compiling the program into a compiled form and then converting the compiled form into a converted form, the converting step including the steps of:

modifying byte code operands from references using identifying strings to references using unique identifiers;

recording all jumps and their destinations in the original byte codes;

converting specific byte codes into equivalent generic byte codes or vice-versa; and

renumbering byte codes in a compiled format to equivalent byte codes in a format suitable for interpretation; and

an interpreter operable to interpret such an application derived from a program written in a high level programming language format; and

a processor coupled to the memory, the processor configured to use the interpreter to interpret the application for execution and to use the communicator to communicate with the terminal.

D7

65

Please Cancel Claim 138 without prejudice.

139. (Amended) A method for use with an integrated circuit card and a terminal, comprising:

storing an interpreter operable to interpret programs derived from programs written in a high level programming language and an application derived from a program written in [a high level]the Java programming language format in a memory of the integrated circuit card wherein the application is derived from a program written in [a high level]the Java programming language format by first compiling the program into a compiled form and then converting the compiled form into a converted form, the converting step including modifying byte code operands from references using identifying strings to references using unique identifiers; and

using a processor of the integrated circuit card to use the interpreter to interpret the application for execution; and

using a communicator of the card when communicating between the processor and the terminal.

~~141~~ 86. (Amended) An integrated circuit card for use with a terminal, comprising:

a communicator configured to communicate with the terminal;

a memory storing:

applications, each application derived from applications having a high level programming language format, and

an interpreter operable to interpret applications derived from applications having a high level programming language format wherein the application is derived from a program written in a high level programming language format by first compiling the program into a compiled form

D8

66

and then converting the compiled form into a converted form, the converting step including the steps of:

modifying byte code operands from references using identifying strings to references using unique identifiers;

recording all jumps and their destinations in the original byte codes;

converting specific byte codes into equivalent generic byte codes or vice-versa; and

renumbering byte codes in a compiled format to equivalent byte codes in a format suitable for interpretation; and

a processor coupled to the memory, the processor configured to:

a.) use the interpreter to interpret the applications for execution,

b.) use the interpreter to create a firewall to isolate the applications from each other, and

c.) use the communicator to communicate with the terminal.

Please cancel Claim 142 without prejudice.

143. (Amended) A microcontroller operable to execute derivative programs which are derivatives of programs written in [an interpretable]the Java programming language having a memory and an interpreter, the microcontroller comprising:

the microcontroller operating within a set of resource constraints including the memory being of insufficient size to permit interpretation of programs written in the [interpretable]Java programming language; and

the memory containing an interpreter operable to interpret the derivative programs written in the derivative of the [interpretable]Java language wherein a derivative of a program written in the [interpretable]Java programming language is derived from the program written in the [interpretable]Java programming language by mapping strings to identifiers.

Claim 144 is not amended herein, but reproduced here for the Examiner's convenience.

~~144~~ 87. A microcontroller comprising:

a memory storing:

a derivative application derived from an application having a class file format wherein the application is derived from an application having a class file format by first compiling the application having a class file format into a compiled form and then converting the compiled form into a converted form, the converting step including:

recording all jumps and their destinations in the original byte codes;

converting specific byte codes into equivalent generic byte codes or vice-versa; and

renumbering byte codes in a compiled format to equivalent byte codes in a format suitable for interpretation, and

an interpreter configured to interpret applications derived from applications having a class file format; and

a processor coupled to the memory, the processor configured to use the interpreter to interpret the derivative application for execution.

REMARKS

This Amendment is intended to be a complete response to the final Office Action of November 22, 2000, and the application is believed to be in condition for allowance. Accordingly, reconsideration is respectfully requested.

68